计算机科学1995 Vol. 22 №. 2

讲授并行程序设计课程的一次实验*)

李晓明
(哈尔滨工业大学计算机系　哈尔滨150006)

摘　要　Parallel Programming, as a selective graduate course, was first offered in Harbin Institute of Technology in the fall semester of 1994. The design, implementation, and feedback of the course is discussed in this article. It is believed that the time has come for an experiment supported parallel programming course offered in our universities.

关键词　parallel processing, parallel programming, curriculum.

一、引言

随着诸如 IBM、DEC 等各大计算机公司纷纷开始研制并行计算机产品，并行计算机将进入计算设备的主流已成必然。这一看法不仅早在学术界，而且也已在工业界达成了共识[1]。高等学校，作为培养人才的地方，如何有效地适应这种形势，是值得考虑的一个问题。不少学校(包括我们学校)，在若干年前，就开设了有关并行处理的研究生课程，主要讲授并行处理的原理和并行算法设计与分析等内容。这无疑对培养我们的研究生，从事有关的科研课题是有帮助的。然而，似乎没有学校开设"并行程序设计"课程，也许是由于条件还不成熟(程序设计课程是要有编程实验配合的)，或者是认为没有必要。

基于如下认识：(1)并行处理的时代已为时不远，(2)并行程序设计是实现并行处理的基本手段，(3)与通常的串行程序设计相比，并行程序设计有很强的特殊性，值得通过一个课程来学习；以及并行程序设计语言 Multi-Pascal 及其模拟环境的发表[2,3]，今年秋我们首次开出了研究生选修课"并行程序设计"，共有8名学生选这门课。现在，课已上完，作为一次实验，对课程的执行情况报告如下。

二、Multi-Pascal 简介

如上所述，程序设计课程要有实验环节配合，并行程序设计的实验要有一个并行程序设计语言和支持用该语言写成的程序的执行环境。真正的并行计算机现在还是个奢侈品，一般学校都没有；就是有也不能让学生随便用。因此我们只能借助于能在串行机上运行的模拟环境。B. Lester 博士研制的 Multi-Pascal 就是一个非常合适的系统[2,3]。

Multi-Pascal 是一种在 Pascal 基础上扩充而得的并行程序设计语言，除一般 Pascal 的基本语言成份外，还提供如下用于并行处理的附加设施：

(1)FORALL 和 FORK 语句，用于创建并发进程。

(2)锁变量 SPINLOCK，通道变量 CHANNEL，用于共享变量的互斥操作，处理器间的通信同步。

Lester 博士同时还提供了一个能模拟执行用 Multi-Pascal 写成的程序的环境 Multi，可以在 PC 或工作站上运行。在这个环境中，程序员可以通过 ARCHITECTURE 说明语句来指明程序将在什么样的机器结构上运行；Multi 能模拟共享存储系统，以及分布存储系统的 RING，MESH，HYPERCUBE，TORUS 等7种互连结构。程序员还可以指明系统的配置，即处理器数。在这样的环境中执行一个程序，Multi 给出的不仅是程序所要求的输出结果，而且还给出和该程序的串行执行相比所得到的加速比等信息。

Multi 还有一个简捷方便的交互式调试和程序运行管理机构，用户可用以设置互连网络的延迟参数，网络是否阻塞等条件，还可以观察所模拟的并行系统中各处理器的忙闲状态剖视图(PROFILE)。

最后，Multi 已是一个相当成熟的系统，运行稳定可靠，足以作为一个教学实验工具。

为给读者一点感性认识，下面是两个 Multi-Pascal 程序及其在 Multi 系统中的运行结果。

第一个程序要求在由4个处理器构成的共享存储系统上求一个数组各元素的平方根。

```
PROGRAM Parallel Squareroot;
ARCHITECTURE SHARED(4);
```

*)本文工作受国防科工委8.5预研基金资助。李晓明　教授。

```
VAR A:ARRAY[1..100]OF REAL;
  I:INTEGER;
BEGIN
  FORALL I:=1 TO 100 DO
    A[I]:=SQRT(A[I]);
END.
```

程序本身没要求输出，Multi 给出的输出是：

```
SEQUENTIAL EXECUTION TIME:1107
PARALLEL EXECUTION TIME:921
SPEEDUP:1.20
NUMBER OF PROCESSORS USED:4
```

第二个程序是在由16个处理器构成的4维立方体网络上求 $\sum i^2$。

```
PROGRAM Summation;
ARCHITECTURE HYPERCUBE(4);
CONST numproc=16;
VAR sum,i:INTEGER;
    ch:ARRAY [0..numproc-1]OF CHANNEL
    OF INTEGER;
PROCEDURE Compute (n, myindex: INTEGER;
VAR sum:INTEGER);
VAR term,position:INTEGER;
    BEGIN
      term:=(myindex+1)*(myindex+2)*
      (myindex+3);
      position:=1;
      WHILE(myindex DIV position MOD 2=0)
        AND(position<n) DO
        BEGIN
          term:=term+ch[myindex];
          position:=position*2;
        END;
      IF myindex<>0 THEN
        ch[myindex-position]:=term
      ELSE
        sum:=term;
      END;

      BEGIN
        FORALL i:=0 TO numproc-1 DO
          (@ i PORT ch[i])Compute(numpro-
          c,i,sum);
          WRITELN(sum);
      END.
```

运行后，Multi 除给出程序要求的输出23256外，还给出：

```
SEQUENTIAL EXECUTION TIME:1381
PARALLEL EXECUTION TIME:570
SPEEDUP:2.42
NUMBER OF PROCESSORS USED:16
```

这两个程序运行的加速比都不高是有道理的。

三、课程的设计和执行

鉴于这是首次开出的（有试验的味道）、实践性很强（要求有实验配合）、本专业研究生（学生已有相当的串行程序设计经验）选修课（学生人数不会很多），在课程结构的设计上我们考虑讲课时数不要太多，但要安排强度足够的程序设计实验。于是计划课程按如下结构进行：

6次共12小时讲课，每周一次，每次课后留一个课外题（PROJECT），在下次上课前完成。在每两次课后安排一次1小时的考问（QUIZ），内容为在计算机上实际编写并调试一个程序。学生的成绩将按如下方式计算：每个 PROJECT 值9分，每次 QUIZ 值17分，最多不超过100分。

课程内容基本参照[2]，6次课的内容包括并行处理及并行计算机系统的基本概念，不同形式的并行计算方式（RELAXED PARALLELISM、SYNCHRONOUS PARALLELISM，PIPELINED PARALLELISM，REPLICATED WORKERS），支持并行程序设计的语言成份的有效应用，以及体系结构、通信参数对并行处理效果的影响等，要学生完成的6个 PROJECT 和3个 QUIZ 是：

Project 1　有序表的归并（共享存储系统）
Project 2　用筛法求素数（共享存储系统）
Project 3　高斯消去法（共享存储系统）
Project 4　在 MESH 结构上的数值积分
Project 5　在全互连结构上实现高斯消去法
Project 6　在 HYPERCUBE 上实现图象处理的区域增长算法
Quiz 1　梯形法数值积分（共享存储系统）
Quiz 2　在 HYPERCUBE 上求和
Quiz 3　在共享存储系统上实现网络的一对多最短路径算法

为每个学生总共安排了40小时的机时来完成这些工作。另外，考虑到学生对程序设计的概念已经很熟，课堂讲授的内容在总体感觉上不会很陌生，讲课用英语进行。

四、总结和心得

按照上述设计执行，八名学生的成绩分别如下，其中 Pi，Qi 分别表示第 i 次 Project 和 Quiz。例如第4个学生做第3个 Project 只得了5分，而他做第1个 Quiz 得了17分。

学生成绩一览表

	P1	P2	P3	P4	P5	P6	Q1	Q2	Q3	总分
1	9	9	9	9	5	9	9	9	17	85
2	9	9	9	9	9	9	9	17	17	97
3	9	9	9	9	9	9	5	9	17	85
4	9	9	5	9	9	0	17	17	17	92
5	5	5	5	5	9	9	5	9	17	69
6	9	9	9	9	5	0	9	9	17	76
7	0	9	5	5	5	5	17	9	17	72
8	0	9	9	9	9	9	9	9	17	80

课上完后，也对学生做了一个调查问卷，由学生对本课程的情况评分，见下表。表中的数字代表为对

应的观点打相同分数的人数。例如对于本课程的必要性，8个同学是一致认同的态度；大部分同学更愿意用英语上此课。

八名学生对课程反映的调查统计情况

	1	2	3	4	5
一、本课程对有效地学习并行程序设计是有必要的					8
二、每次课的内容太多	4	2	1	1	
三、PROJECT 作业对掌握有关内容是有帮助的				1	7
四、所安排的机时不够完成 PROJECT	1		3	1	3
五、小测验太难了	2	1	3	2	
六、最好用汉语上课	4	2		2	

作为任课教师，有如下几点体会：

(1)并行程序设计是一个非常引人入胜的活动。我们不仅要关心程序的正确性，还要在程序的运行效率上投入较大的注意力。通常一个程序8个人写出8个结果；程序都是对的，但加速比可能有很大差别。例如最后一个 QUIZ，有的程序的加速比可以达到10以上，而有的还不能超过3(不论用多少处理器)。

(2)讲课时数还应稍多一些，例如16小时，就可以覆盖关于"计算-汇集-广播模型"以及 REPLICATED WORKER 模型中的程序终止检测等内容。

(3)每周只安排一次课是重要的，否则学生没时间完成作业。

(4)在一个小时里要完成哪怕是很简单的程序也是困难的。从上述 QUIZ 得分情况可以看出这一点。第三个 QUIZ 大家都做得比较好是因为事先将有关内容大致通报给了学生，请他们有所准备。

致谢 谨在此特别感谢加拿大 Concordia 大学的陶立新博士，如果不是他向我们介绍 Dr. Lester 的书，上述工作都将不会发生。

参考文献

[1] Henry Burkhardt, The Smoke is clearing, and parallel processing has won, IEEE Spectrum, Jan. 1994, page 49

[2] Bruce P. Lester. The Art of parallel Programming. Englewood Cliffs, NJ: Prentice Hall Inc., 1993

[3] 刘宏伟，李晓明，Multi-Pascal：一个经济有效的并行程序设计研究工具，哈尔滨工业大学计算机系技术报告，PACT-TR-94-025。

(上接第22页)

PCODE 操作要求它的操作数有特定类型(象 ADD)，同步需不同处理；对 FORK/JOIN 和产生/消费型可引入 REPLACE_IF_EQ 原语，FUTURE 需特别处理；对副作用型可引入(WAIT S)，(SIGNAL S)，(SUSPEND F)和(ACTIVE S)原语。任务调度策略是先满足某个任务，其次是别的，这是一个不公平的调度。废料收集的算法是基于复制和 BACKER 增量型废料收集的，在分布系统中它的实现更复杂。

※ ※ ※ ※ ※

鸣谢 借此机会感谢赵银亮博士，党华锐和 Abdulrazaque Memon。

参考文献

[1] Burton F. W., Annotations to control parallelism and reduction order in the distributed evaluation of functional progarams, Transactions on Progress in Language and Systems 6(2)159-174(1984)

[2] Burton F. W., Functional programming for Concurrent and Distributed Computing, The Computer Journal 30(5)437-450(1987)

[3] Robert H. Halstead, JR., MULTILISP: A Language for Concurrent Symbolic Computation, ACM Transactions on Programming Languages and Systems, 7(4)501-538(1985)

[4] Peyton Jones S. L., Parallel Implementations of Functional Programming languages, The Computer Journal 32(2)175-186(1989)

[5] Kenneth H. et al., The Philosophy of Lisp, Commun. ACM 34(9)40-47(1991)

[6] Layer D. K. and Richardson C., Lisp Systems in the 1990s, Commun. ACM 34(9) 48-57 (1991)

[7] Baker H., Actor Systems for Real-Time Computation, Tech. Rep. Tr-197, Massachesetts Institute of Technology, Laboratory for Computer Science, Cambridge, Mass., Mar. 1978